Object-Oriented Programming

Programming Project

Academic Year 2025-2026

This project can be completed individually or by pairs of students, and must be submitted by email to `bvergain@uliege.be` *for April 19th at the latest. Projects returned after the deadline will not be graded. Except for classes present in the Java Class Library, which you are allowed to use, all the code that you submit must have been written by yourselves. Fraud will be prosecuted according to the policy of ULiège (cf.* `https://www.student.uliege.be/cms/c_11187649/`*). This include copying or adapting code obtained from other students, from online sources, or generated with the help of AI tools.*

The project consists in writing a Java program that is able to solve *jigsaw puzzles*, whose aim is to assemble a given set of pieces into a rectangle.

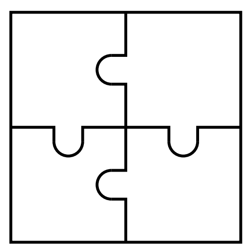

Figure 1: Example of jigsaw puzzle

In the jigsaw puzzles that we consider, the pieces have a square shape, with each side being either flat (F), with a bump (B), or with a pit (P). Like illustrated in Figure 1, a puzzle solution is valid if the flat sides correspond to the outside boundaries of the enclosing rectangle, and the sides with bumps and pits match each other inside of this rectangle.

The pieces that compose a given puzzle are described by a specification file, the format of which is imposed. This file takes the form of a text file in which line separators are either line feeds (`"\n"`) or carriage returns followed by line feeds (`"\r\n"`). This file contains, in this order:

- a line `"width height"` specifying the horizontal and vertical sizes of the final assembled rectangle, expressed in numbers of pieces.
- `width` $\times$ `height` lines that each specify the shape of an individual piece of the jigsaw puzzle. Each such line contains four consecutive characters `'F'`, `'B'` or `'P'` describing the form of the successive sides of the piece, in clockwise order starting from the top side.

For instance, the following specification file describes the elements of the puzzle illustrated in Figure 1.

```
2 2
FFBP
BFFP
BBFF
PPFF
```

Your program must accept a single command-line argument providing the name of the specification file to be read. It should parse this file, solve the corresponding jigsaw puzzle, and display a solution. In the case of any error, such as an invalid specification file, or if it cannot find a solution, the program must display an error message indicating clearly the nature of the problem.

If a solution is found, it must be displayed by enumerating row by row from top to bottom, and from left to right within each row, the pieces that compose it. Each piece is described by two numbers corresponding to its index in the specification file (the first piece is numbered 1, the second 2, and so on), and its orientation expressed as the number of clockwise quarter turns (from 0 to 3) that it has been rotated from its reference position expressed in the specification file.

For example, the solution in Figure 1 can be displayed as follows:

```
2 2
3 2
4 0
1 1
```

Important guidelines

- The main class of your project must be named `Puzzle`. Your program must expect to be invoked with a single argument containing the name of the specification file.

  Examples of correct specification files will be provided on the web page of the course. You are strongly encouraged to check that your program works as expected on those particular examples.
- Your program must only receive input data from the specification file and the command-line argument, and only output data in text form on the console. You are not asked to develop a graphical interface.
- Your submission must take the form of a `.zip` or `.tar.gz` archive containing the Java source code of your project. The name of this archive must be formed by the student IDs of the members of your team separated by a dash symbol, for instance `"s2312345-s2054321.zip"`. This file must be attached to an email bearing the subject `"OOP project submission"`.
- After extracting the contents of your submitted archive in the current directory, it should be possible to compile your project on a Linux system with the command

  ```
  javac be/uliege/montefiore/oop/*.java
  ```

  (Note that if you develop under another operating system, or within an IDE, the syntax of this command can slightly differ.)
- Your project will be evaluated with Java 8. It is thus important to make sure that your submission does not rely on features introduced in later versions of the Java language.
- The evaluation will take into account not only the correct implementation of the requirements of this problem statement, but also the compliance of your source code to the principles of object-oriented programming, as well as its modularity, efficiency, readability, simplicity, and portability.